AFRL-IF-RS-TR-2000-52
Final Technical Report
April 2000

QUERY RELAXATION AND ITS APPLICATIONS IN INFORMATION SYSTEMS

University of California, Los Angeles

Wesley W. Chu

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

AFRL-IF-RS-TR-2000-52 has been reviewed and is approved for publication.

APPROVED: Raymond A. Liuzzi

RAYMOND A. LIUZZI
Project Engineer

FOR THE DIRECTOR: [illegible]

NORTHRUP FOWLER, Technical Advisor
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE APRIL 2000	3. REPORT TYPE AND DATES COVERED Final Apr 99 - Mar 00

4. TITLE AND SUBTITLE QUERY RELAXATION AND ITS APPLICATIONS IN INFORMATION SYSTEMS	5. FUNDING NUMBERS C - F30602-99-2-0526 PE - 62232N PR - R427 TA - 00 WU - P2
6. AUTHOR(S) Wesley W. Chu	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of California, Los Angeles Department of Computer Science 405 Hilgard Avenue Los Angeles CA 90095	8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/IFTD 525 Brooks Road Rome NY 13441-4505 — SPAWARSYSCEN D44209 53425 Patterson Road San Diego CA 92152-7151	10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2000-52

11. SUPPLEMENTARY NOTES
Air Force Resesearch Laboratory Project Engineer: Raymond Liuzzi/IFTD/(315) 330-3577
SPAWASYSCEN Project Engineer: Leah Wong/D44209/(619) 553-4127

12a. DISTRIBUTION AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The goal of this project is to transfer the Cooperative Database (CoBase) technology to the Navy/SPAWAR and AFRL/Rome. CoBase was installed at the SPAWAR computing environment to perform query relaxation in an agent-based architecture. CoBase serves as an agent and communicates with other agents via the FIPA knowledge communication protocol. A customized interface to fit the SPAWAR environment, which includes building a JDBC gateway interface and developing a special start-up script was performed. The CoBase Content Language (CCL) was developed to support agent communication. Type Abstraction hierarchies (a knowledge-base) were generated for the SPAWAR database for query relaxation. Demo queries showcased CoBase's relaxation and relaxation control capabilities. CoBase was also installed at the AFRL facility located in Rome NY. A limited demonstration of CoBase capabilities was performed at AFRL/Rome.

14. SUBJECT TERMS Cooperative Rules, Cooperative Operators, Rule Relaxation, Active Database Systems, Cooperative Active Database Systems	15. NUMBER OF PAGES 48
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Executive Summary

The goal of this project is to transfer the Cooperative Database (CoBase) technology to the Navy operating environment via two demonstrations. The first demo is to install CoBase at the SPAWAR computing environment with the Navy database and scenario. The second demo is to have CoBase perform query relaxation in an agent-based architecture. In this case, CoBase serves as an agent and communicates with other agents via the FIPA knowledge communication protocol.

To operate CoBase at SPAWAR, we have to customize the interface to fit the SPAWAR environment, which includes building a JDBC gateway interface and developing a special start-up script (section B). The CoBase Content Language (CCL) is developed to support agent communication (section C). Type Abstraction Hierarchies (a knowledge-base) are generated from the Navy database (section E) for query relaxation. The TAHs are then installed at CoBase at SPAWAR. The demo queries showcase CoBase's relaxation and relaxation control capabilities (section D).

Both of the demos were successfully accomplished at SPAWAR. The project has been successfully completed. We have also identified areas for further improvements (section G). Some of them have already been implemented (e.g. the change from the COBRA to Socket for communicating with JAVA GUI interface). We have also gained experience in software installation at different environments, which resulted in the development of an installation shell that centralizes all the environment variables. This will greatly ease future CoBase distribution.

A. The CoBase Technology

We shall first describe the query relaxation process followed by the innovative knowledge representation of the type abstraction hierarchy (TAH) and its automatic generation from data sources. Finally, we will present two query relaxation examples.

A.1. Query Relaxation

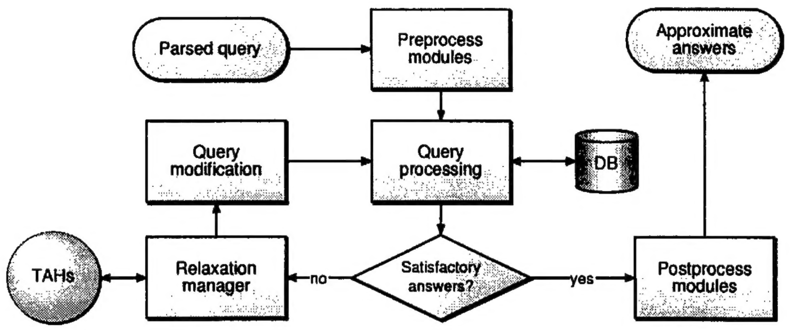


Figure 1 Flow Chart for Processing CoBase Queries.

In conventional databases, if required data is missing, if an exact answer is unavailable, or if a query is not well formulated with respect to the schema, the database returns a null answer or an error. An intelligent system would be much more resourceful and cooperative by permitting conceptual level queries (containing concepts that may not be expressed in the database schema) when the user does not know the precise schema and provide approximate answers when some data is missing, or even volunteering to the user associative (relevant) information to the query answer.

The cooperative database system, CoBase [CMB93, CYC+96], provides *approximate and conceptual query answers* that *approximately* match query conditions when exact answers are unavailable. The knowledge base used by CoBase is represented by a tree structure hierarchy called the Type Abstraction Hierarchy (TAH) [CC92]. Using such a representation, CoBase is able to provide an efficient, scalable, and structured technique to relax attribute values incrementally. Such relaxation can be controlled by the user or through defaults provided by the user model.

Figure 2 depicts the data flow of query relaxation. When a query is presented to CoBase, the pre-processor first relaxes any explicit cooperative operators (e.g., approximate, near-to, similar-to) in the query. The modified query is then presented to the underlying database system for execution. If no answers are returned, then the *relaxation manager* uses the type abstraction hierarchies to iteratively perform query relaxation. The policy for relaxation control depends on many factors, including the user's profile and query context. The list of relaxation control operators includes attribute relaxation order (e.g., *size* first, then *location*), non-relaxable attribute(s), the number of answers required, etc. The relaxation manager combines these control policies to generate approximate answers that best match the user's original query. Finally, a post-processing module filters and ranks the answers according to a given error measure (e.g., mean-square error).

A.2. Type Abstraction Hierarchies (TAHs)

Type Abstraction Hierarchies (TAHs) provide multi-level classification of instances within a specific domain [CC94]. TAH nodes can be labeled by conceptual terms that describe the range of values covered by their children nodes. In Figure 3, for example, the *Medium-Range* (*i.e.*, from 4,000 to 8,000 ft.) in the TAH for *runway-length* is a more abstract (conceptual) representation than a specific runway length in the same TAH (*e.g.*, 6,000 ft). Likewise, *SW Tunisia* is a more abstract (conceptual) representation than individual airports (*e.g.*, Gafsa). A higher-level and more abstract object representation corresponds to multiple lower-level and more specialized object representation. Querying an abstractly represented object is equivalent to querying multiple specialized objects.

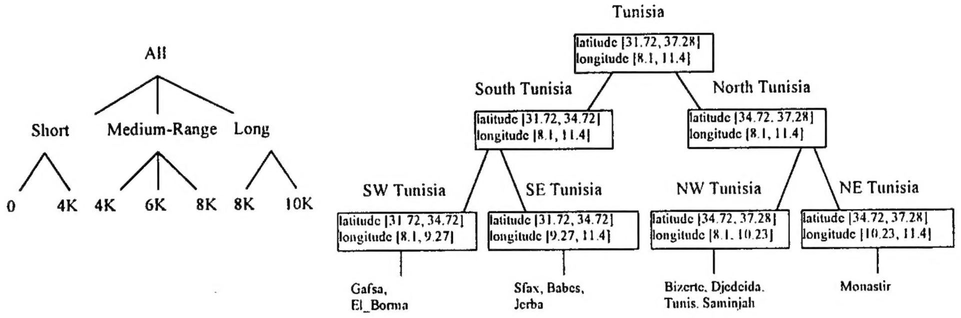


Figure 2 Type Abstraction Hierarchies.

Unmatched attribute values or concepts can be relaxed and approximately matched by moving up the type abstraction hierarchy (generalization) to broaden the search scope, and then moving down the type abstraction hierarchy (specialization) to find an approximate match. The quality of the approximations to the exact match, known as the relaxation error, can be computed and returned to the user [CC94]. Many objects require multi-dimensional representation. For example, the location of an object on a 2-D plane is represented by two attributes, x and y. Therefore, we have extended TAHs to classify multiple attributes (i.e., database tuples) capturing generalization and specialization between multi-dimensional data instances [CCH+96].

TAHs can capture different types of knowledge, such as database attribute values, image features, and text pattern features. Thus, TAHs can be generated from different information sources, such as structured databases, image feature databases, and ontological terms used in resource registration.

A.3. Automatic Construction of TAHs from Information Sources

TAHs can be generated and maintained automatically based on the data instances. For numerical attribute values, our Distribution Sensitive Clustering algorithm (DISC) method considers both *frequency* and *value* distributions of data, making the discovered concepts more content sensitive [CCH+96]. To optimize the quality of generated hierarchies, DISC partitions the data set of one or more attributes selected by the user into clusters such that the average *relaxation error* (the value difference between the instance value and the target value) is minimized [CCH+96]. For non-numerical attribute values, our Pattern Based Knowledge Induction (PBKI) method first derives IF-THEN rules for all pairs of attributes. Based on the derived rules, PBKI then computes co-occurrences between all pairs of values for a single attribute [MC93]. Starting with each value as a cluster, PBKI recursively merges clusters with the highest co-occurrences to form larger clusters. The merging process is terminated when there is only one cluster left.

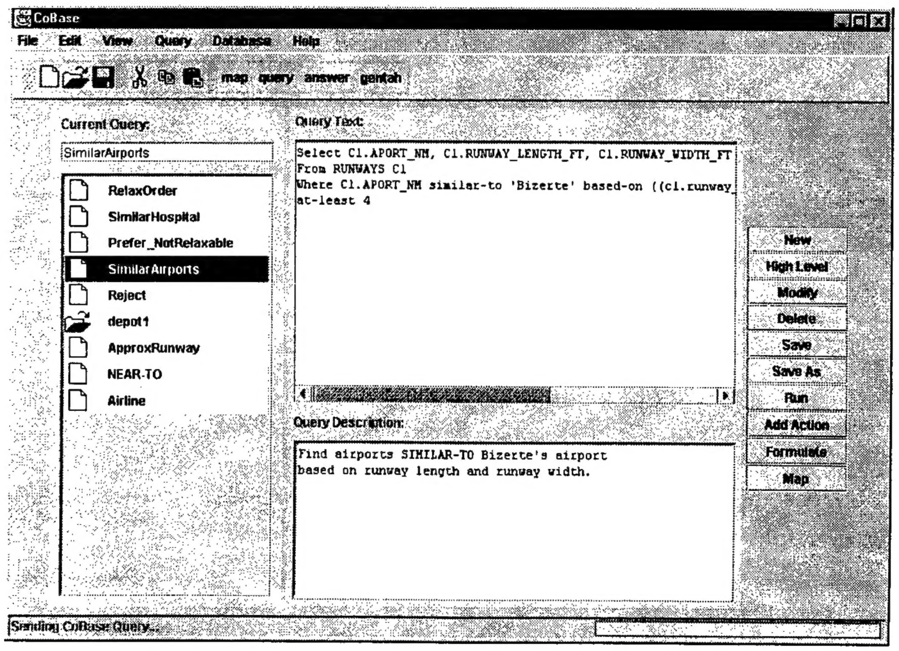
Both DISC and PBKI have polynomial time complexity, making them scalable to large application domains. To demonstrate the effectiveness of DISC and PBKI, we have used them to generate TAHs for a large transportation database consisting of 94 relations, the largest one of which has 12 attributes and 195,598 tuples. The results show that our methods are efficient, taking less than a minute on a Sun SPARC 10 Workstation to generate TAHs for all but a few cases with a large number of tuples. The remaining cases will require less than 20 minutes.

A.4. Query Relaxation Examples

We shall present two sample queries to illustrate the query relaxation technology of CoBase. The first query, Q1, illustrates the "similar to" and "at least" constructs as shown in Figure 4a. The TAH used for the relaxation of the runway length is shown in Figure 4b and the results are shown in Figure 4c. The second query, Q2, is an airline-ticketing query to buy a ticket from Los Angeles to New York or Newark with departure time, arrival time, and cost as constraints, and choice of airline and stop over city as preferences. The Extended SQL (CoSQL) query used in CoBase is shown in Figure 5a and the results are shown in Figure 5b.

Q1: Similar-To Query

This query finds airports similar to Bizerte airport based on runway length and runway width.



```
Select C1.APORT_NM, C1.RUNWAY_LENGTH_FT, C1.RUNWAY_WIDTH_FT
From RUNWAYS C1
Where C1.APORT_NM similar-to 'Bizerte' based-on ((c1.runway_
at-least 4
```

```
Find airports SIMILAR-TO Bizerte's airport
based on runway length and runway width.
```

Figure 3a The Similar to Query.

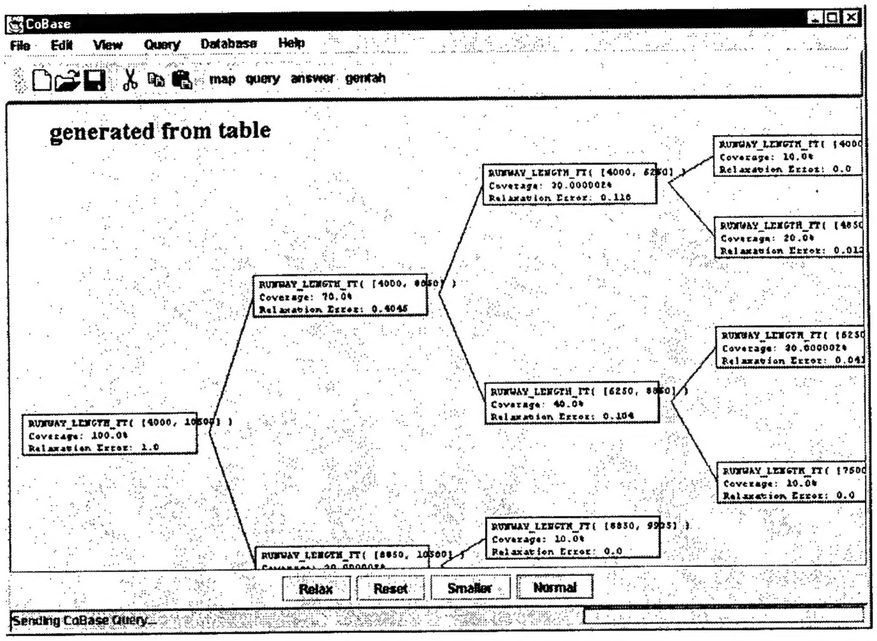
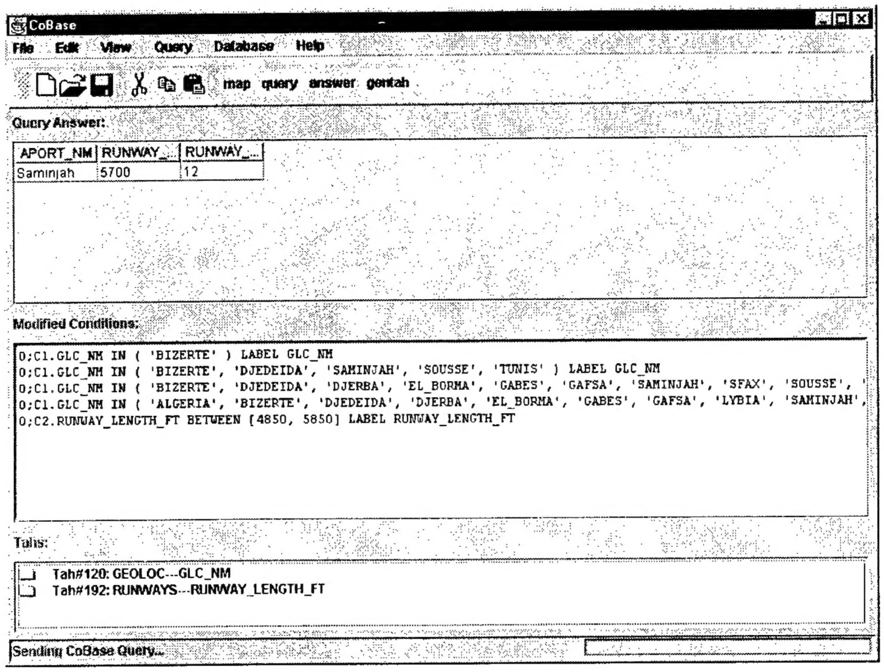


Figure 3b The TAH used for relaxation of runway length to find similar airports.



Query Answer:

APORT_NM	RUNWAY_...	RUNWAY_...
Saminjah	5700	12

Modified Conditions:

```
0;C1.GLC_NM IN ( 'BIZERTE' ) LABEL GLC_NM
0;C1.GLC_NM IN ( 'BIZERTE', 'DJEDEIDA', 'SAMINJAH', 'SOUSSE', 'TUNIS' ) LABEL GLC_NM
0;C1.GLC_NM IN ( 'BIZERTE', 'DJEDEIDA', 'DJERBA', 'EL_BORMA', 'GABES', 'GAFSA', 'SAMINJAH', 'SFAX', 'SOUSSE', '
0;C1.GLC_NM IN ( 'ALGERIA', 'BIZERTE', 'DJEDEIDA', 'DJERBA', 'EL_BORMA', 'GABES', 'GAFSA', 'LYBIA', 'SAMINJAH',
0;C2.RUNWAY_LENGTH_FT BETWEEN [4850, 5850] LABEL RUNWAY_LENGTH_FT
```

Tahs:

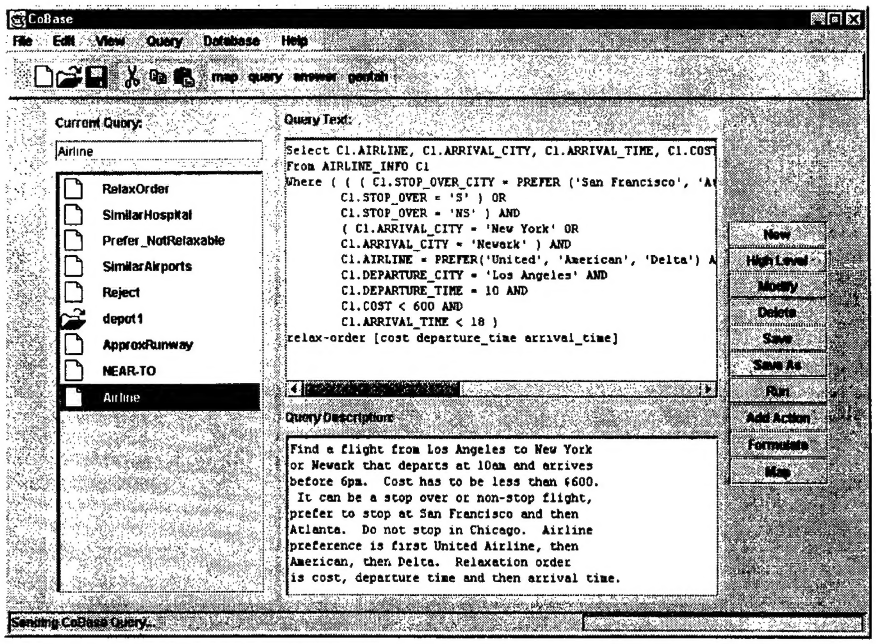
- Tah#120: GEOLOC---GLC_NM
- Tah#192: RUNWAYS---RUNWAY_LENGTH_FT

Sending CoBase Query...

Figure 3c Answers to the Similar to Query.

Q2: Airline Ticketing Query

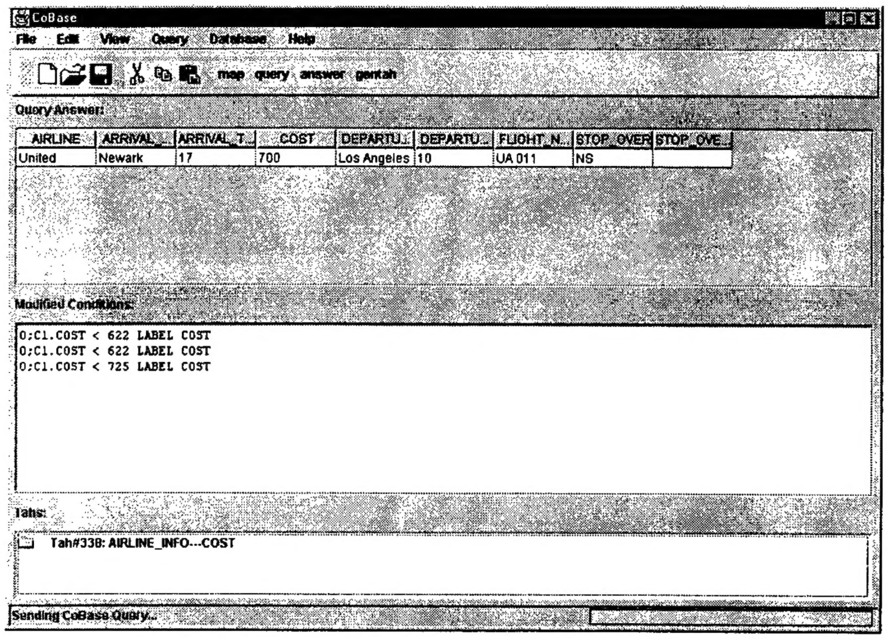
This query shows relaxation control on costs, arrival time, departure time, preference on airlines, and stop over locations. When the relaxation control orders changes (e.g., arrival time, departure time, and costs), different answers are obtained.



```
Select C1.AIRLINE, C1.ARRIVAL_CITY, C1.ARRIVAL_TIME, C1.COST
From AIRLINE_INFO C1
Where ( ( ( C1.STOP_OVER_CITY = PREFER ('San Francisco', 'At
        C1.STOP_OVER = 'S' ) OR
        C1.STOP_OVER = 'NS' ) AND
        ( C1.ARRIVAL_CITY = 'New York' OR
        C1.ARRIVAL_CITY = 'Newark' ) AND
        C1.AIRLINE = PREFER('United', 'American', 'Delta') A
        C1.DEPARTURE_CITY = 'Los Angeles' AND
        C1.DEPARTURE_TIME = 10 AND
        C1.COST < 600 AND
        C1.ARRIVAL_TIME < 18 )
relax-order [cost departure_time arrival_time]
```

Find a flight from Los Angeles to New York or Newark that departs at 10am and arrives before 6pm. Cost has to be less than $600. It can be a stop over or non-stop flight, prefer to stop at San Francisco and then Atlanta. Do not stop in Chicago. Airline preference is first United Airline, then American, then Delta. Relaxation order is cost, departure time and then arrival time.

Figure 4a Airline Ticketing Query.



AIRLINE	ARRIVAL_...	ARRIVAL_T...	COST	DEPARTU...	DEPARTU...	FLIGHT_N...	STOP_OVER	STOP_OVE...
United	Newark	17	700	Los Angeles	10	UA 011	NS	

```
0;C1.COST < 622 LABEL COST
0;C1.COST < 622 LABEL COST
0;C1.COST < 725 LABEL COST
```

Figure 4b Relaxed result of the Airline Ticketing Query.

B. SPAWAR CoBase Operating Environment

B.1. CoBase Setup at SPAWAR

There are 2 client programs written in JAVA. One is a query wizard GUI and the other is a command-line client program. To talk to the CoBase server, the Query Wizard GUI uses the CORBA interface and the command-line client program uses the socket interface.

Similarly, there are 2 CoBase server programs written in C++. One has a CORBA interface and the other has a socket interface.

Since SPAWAR does not use the ODBC driver for their Oracle database, we developed a JDBC driver. To do that, we had to run a middleware-like server that ran between the CoBase server and JDBC driver as shown in Figure 5.

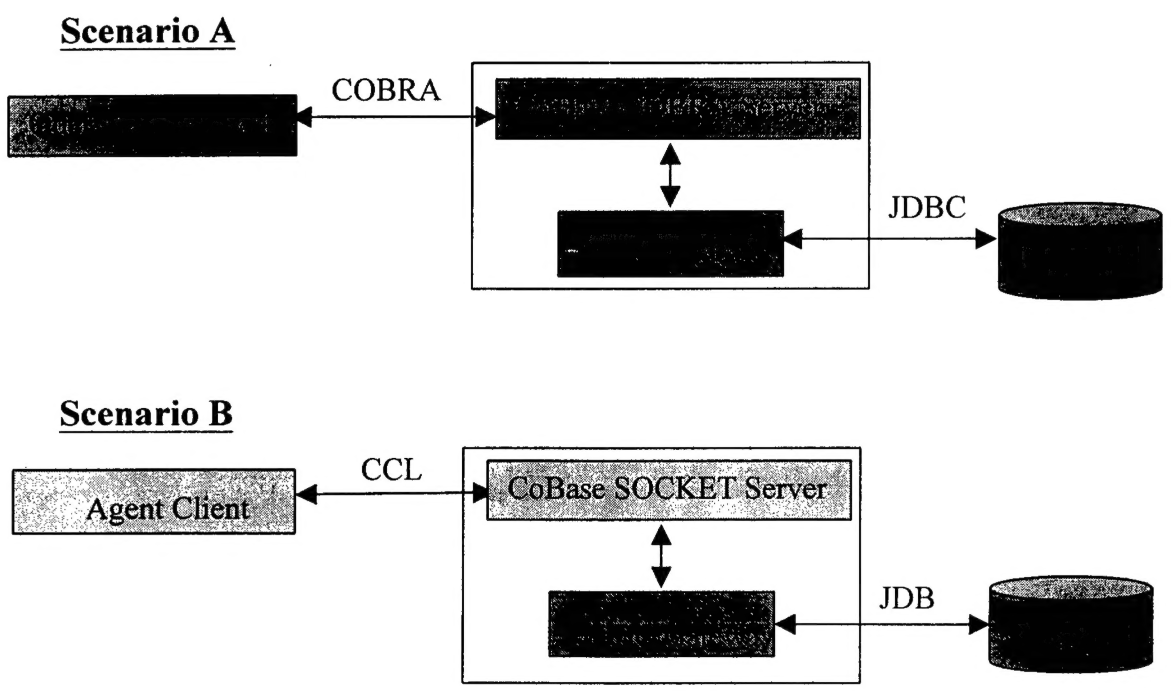


Figure 5 CoBase Operating Environment at SPAWAR. The CoBase GUI requires CoBaseCorbaServer, while the client that we provided requires CoBaseSocketServer. JDBCSocketServer is shared by both procedures.

B.2. CoBase Start-Up

1. Setup environment properly using one of the provided shell scripts:

```
sh/bash/zsh users :      bash-2.01$ source setenv.sh
csh/tcsh users    :      bash-2.01$ source setenv.csh
```

2. To run GUI, CoBaseCorbaServer needs to be run. This in turn requires ORBIX daemon. First, run "orbixd".

```
bash-2.01$ /net/mrmagoo/disk/Orbix/Orbix2.3/bin/orbixd &
```

Then, the following message will appear:

```
bash-2.01$ [orbixd: Server "IT_daemon" is now available to the network]
[ Configuration TCP/1570/Orbix-XDR ]
```

We already registered the CoBaseCorbaServer program as a nickname of a "CoBaseBroker" under ORBIX daemon. To verify this, type the following:

```
bash-2.01$ catit CoBaseBroker
```

Then it should show the following message:

```
Server details for server  : CoBaseBroker

   Comms            : tcp
   Code             : cdr
   Activation       : shared
   Owner            : ucla
   Launch           : ;all;
   Invoke           : ;all;

      Marker           Launch Command

      *                /home/ucla/demo/CoBaseCorbaServer/test-server
```

When a client sends a request to the program registered as a "CoBaseBroker", the ORBIX daemon intercepts it and launches the program "/home/ucla/demo/CoBaseCorbaServer/test-server" instead. Therefore, the CoBaseCorbaServer doesn't have to be running all the time.

3. To run the JDBCSocketServer for Oracle7 @ sarx. Go to JDBCSocketServer subdirectory and type:

```
bash-2.01$ test-server
```

This will run the gateway program.

4. Now go to "GUI" directory and type:

```
bash-2.01$ ./run
```

Then it will run the Java query wizard program.

C. CoBase Content Language (CCL)

C.1. Syntax

For the CoBase socket server to communicate with other agents via a socket connection, a customized content language specific to CoBase, called CCL (CoBase Content Language), was devised. CCL follows the FIPA (as well as its predecessor KQML) communication procedural format, which transfers data based on textual representation. The sending object is represented in the following format for sending objects:

```
{data-type {value}}
```

The `data-type` represents the type of the object, and the `value` is the string representation of the object. For instance, a list of TAH names could be represented as `{tah_names {{Tah#100} {Tah#210} {Tah#122}}}`, and a CoSQL query could be represented as `{cosql_query {SELECT APORT_NM FROM RUNWAYS WHERE APORT_NM = `BIZERTE'}}`.

The following is a list of data types supported by the CoBase Content Language:

`cosql_query`
This data type represents a query in the CoSQL language.

```
{cosql_query {SELECT APORT_NM
        FROM RUNWAYS WHERE RUNWAY_LENGTH_FT = 3000}}
```

`tah_name`
This data type is transferred to CoBase as the argument for the `get_list_tah_nodes` function call.

```
{tah_name {Tah#120}}
{tah_name {My Tah Name}}
```

`query_answers`
This data type represents a set of query answers. The entire set of query answers is enclosed with curly braces ({}), with each tuple enclosed in curly braces, and each value within a tuple enclosed in curly braces. This last level of braces is necessary because tuple values may contain text separated by spaces. We will use this general format for any data types that are lists of items.

The answer to the query ``SELECT MAKE, MODEL, COLOR FROM CARS'' may get the following query_answer object:

```
{query_answers {{{PONTIAC} {GRAND PRIX} {GREEN}}  {{HONDA} {ACCORD}
{RED}}  {{TOYOTA} {LAND CRUISER} {MAGENTA}}}}
```

tah_names
This data type represents a list of TAH names. This data type will be returned by CoBase when asked for the list of TAH names. Curly braces are used for each TAH name since TAH names may consist of multiple words.

```
{tah_names {{TAH#120} {AUTOMOBILES TAH}}}
```

modified_conds
This data type represents a list of modified conditions. Modified conditions are the query conditions changed by CoBase during the relaxation process.

```
{modified_conds {{RUNWAY_LENGTH_FT BETWEEN [2000, 4000]}
{RUNWAY_WIDTH_FT [30, 50]}}}
```

C.2. Example

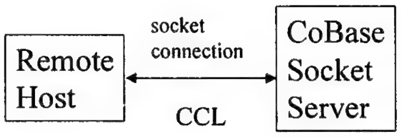


Figure 6: A Client Communicates with CoBase through a Socket.

Figure 6 shows the basic configuration for a remote host that communicates with the CoBase socket server. The requirements for communicating with CoBase through a socket are quite simple. First, the host name and the port number for the CoBase server must be obtained. Using this information, the agent program then must open a socket connection. Once a socket connection has been made, the client program can use CCL to send requests or retrieve information from CoBase, similar to the FIPA and KQML protocols. The major difference between using FIPA and the KQML interfaces and using the socket interface is the format of the sending functions. Since the socket interface does not use the various fields that are available in the FIPA and KQML, CCL function calls use the following format:

$\{\texttt{function-name}\ \texttt{arg}_1\ \texttt{arg}_2\ \ldots\ \texttt{arg}_n\}$

where `function-name` is the name of one of the functions in the CCL, and the $\texttt{arg}_i$'s are the functions arguments. The arguments must be one of the data objects defined in the CCL. Other than these minor differences, information sent to and from CoBase and the client is similar to the KQML message contents. Therefore, CCL can be easily wrapped by the FIPA or KQML by adding additional protocol set-up conditions (e.g. language, ontology, etc.). The following are some examples of using the CoBase socket interface:

Suppose the agent wants to send CoBase a query for execution. The following string will be written by the agent into the socket:

```
{execute_cosql_query {cosql_query
      {select CTRY_CODE, AIRFIELD_NAME, LAT, LON
            from   AIR_RWY
            where  AIRFIELD_NAME similar-to 'MALI'
              based-on ((RWLTH,1.0),(RWWTH,1.0))
            at-least 7}
}}
```

In this example, `execute_cosql_query` is the function name, and the entire `cosql_query` expression is its one argument. In this query, agent is asking "find at least 7 airfield names similar to MALI with respect to its runway length and runway width".

Then, CoBase server finds answers after numerous relaxation and replies with the following string:

```
{query_answers {
        {{ID}{MALI}{-8.1325}{124.5964}{75}{3477}{0}}
        {{ID}{ KASIGUNCU}{-1.4172 }{120.6578}{ 75}{ 3644}{ 0.564031}}
        {{ID}{ PANGSUMA }{.8358}{ 112.9358}{ 75 }{3294}{ 0.61807}}
        {{ID}{ KALIMARU}{ 2.1547}{ 117.4325}{ 78 }{3664 }{0.631661}}
        {{ID}{ LHOK SUKON}{ 5.0694}{ 97.2592}{ 69 }{3675}{ 0.669039}}
        {{ID}{ NANGA PINOH I}{ -.3489}{ 111.7475}{ 75}{ 3272}{ 0.692374}}
        {{ID}{ MUKO MUKO}{ -2.5442}{ 101.0883}{ 75}{ 3232}{ 0.827471}}
        {{TH}{ LOM SAK}{ 16.8208}{ 101.2539}{ 82}{ 3773}{ 1}}
}}
```

D. Queries used in the Demonstration

The following is a list of queries and answers that were demonstrated in the September demo.

Query 1: similar-to based on runway length and width in AIR_RWY table

Find Airfield **similar-to** 'MALI' based on runway length and runway width.

```
file: navy-air-similar-to.qry

"select     CTRY_CODE, AIRFIELD_NAME, LAT, LON
from  AIR_RWY
where AIRFIELD_NAME similar-to 'MALI'
            based-on ((RWLTH,1.0),(RWWTH,1.0))
at-least 7"

transformed SQL query:
SELECT  AIR_RWY.CTRY_CODE, AIR_RWY.AIRFIELD_NAME, AIR_RWY.LAT, AIR_RWY.LON,
AIR_RWY.RWWTH, AIR_RWY.RWLTH
FROM    AIR_RWY
WHERE   AIR_RWY.RWWTH BETWEEN [68, 87]
        AND AIR_RWY.RWLTH BETWEEN [3175, 3832]

      answers WITHOUT relaxation

ID MALI -8.1325 124.5964 75 3477

      answers WITH relaxation

ID MALI -8.1325 124.5964 75 3477 0
ID KASIGUNCU -1.4172 120.6578 75 3644 0.564031
ID PANGSUMA .8358 112.9358 75 3294 0.61807
ID KALIMARU 2.1547 117.4325 78 3664 0.631661
ID LHOK SUKON 5.0694 97.2592 69 3675 0.669039
ID NANGA PINOH I -.3489 111.7475 75 3272 0.692374
ID MUKO MUKO -2.5442 101.0883 75 3232 0.827471
TH LOM SAK 16.8208 101.2539 82 3773 1
```

Query 2: relaxation based on runway length and width in AIR_RWY table

Find at least 5 airfield with runway length >3000 and runway width >300.

```
file: navy-air-runway.qry

"select AIRFIELD_NAME, RWLTH, RWWTH
from    AIR_RWY
where   RWLTH > 3000 and RWWTH > 300
at-least 5",

transformed SQL query:
SELECT  AIR_RWY.AIRFIELD_NAME, AIR_RWY.RWLTH, AIR_RWY.RWWTH
FROM    AIR_RWY
WHERE   AIR_RWY.RWLTH > 2282
        AND AIR_RWY.RWWTH > 191
```

```
      answers WITHOUT relaxation

KALIJATI 3921 328

      answers WITH relaxation

AMPHOE BAN KHWAO 4675 249
MORAWA 5300 300
TYABB 2500 200
ASTRA KESETRA 5703 199
KALIJATI 3921 328
```

Query 3: similar-to based on location in CITY table

Find at least 5 cities nearby 'Dili' based on location in CITY table.

```
file: navy-city-similar-to.qry

"select      CTRY_CODE, CITY_NAME, DESIG, LAT, LON
from  CITY
where CITY_NAME similar-to 'Dili'
            based-on ((LAT,1.0),(LON,1.0))
at-least 5"

transformed SQL query:
SELECT  CITY.CTRY_CODE, CITY.CITY_NAME, CITY.DESIG, CITY.LAT, CITY.LON
FROM    CITY
WHERE   CITY.LON BETWEEN [124.958, 125.833]
        AND CITY.LAT BETWEEN [-14.9584, -6.125]

      answers WITHOUT relaxation

null

      answers WITHOUT relaxation

ID Dili ADM2 -8.6667 125.6667 0
ID Aileu ADM2 -8.7167 125.5667 0.150006
ID Ermera ADM2 -8.7500 125.4167 0.353553
ID Ainaro ADM2 -8.9167 125.5833 0.353596
ID Manofahi ADM2 -8.9833 125.6667 0.42478
ID Likisia ADMD -8.6667 125.3333 0.447321
ID Bobonaro ADM2 -8.9167 125.2500 0.651985
ID Kovalima ADMD -9.3333 125.3333 1
```

Query 4: similar-to based on location in PORTS table

Find at least 3 ports nearby 'DOVER'.

```
file: navy-port-similar-to.qry

"select     CTRY_CODE, NAME, SEC_CLASS_CODE, LAT, LON
from  PORTS
where NAME similar-to 'DOVER'
            based-on ((LAT,1.0),(LON,1.0))
at-least 3"

transformed SQL query:
SELECT  PORTS.CTRY_CODE, PORTS.NAME, PORTS.SEC_CLASS_CODE, PORTS.LAT,
PORTS.LON
FROM    PORTS
WHERE   PORTS.LON BETWEEN [80.3083, 160.133]
        AND PORTS.LAT BETWEEN [-77.85, -43.1167]

      answers WITHOUT relaxation

null

      answers WITH relaxation

AS DOVER U -43.3167 147.0167 0
AS HYTHE U -43.4333 147.0000 0.446945
AS PORT HUON U -43.1667 146.9833 0.583103
AS MIDDLETON U -43.2333 147.2667 1
```

Query 5: approximation & at-least

Find at least 3 airfields whose runway length > 6250 and runway width > 100 for landing a Boeing 707 type aircraft and are located at the approximate 3000 elevation level.

```
file: navy-air-at-least.qry

"select     CTRY_CODE, AIRFIELD_NAME, RWLTH, RWWTH, ELEV
from  AIR_RWY
where RWLTH > 6250 and RWWTH > 100
and   ELEV =~ 3000
at-least 3"

transformed SQL query:
SELECT  AIR_RWY.CTRY_CODE, AIR_RWY.AIRFIELD_NAME, AIR_RWY.RWLTH,
AIR_RWY.RWWTH, AIR_RWY.ELEV
FROM    AIR_RWY
WHERE   AIR_RWY.RWLTH > 6167
        AND AIR_RWY.RWWTH > 94
        AND AIR_RWY.ELEV BETWEEN [2375, 3471]
```

```
      answers WITHOUT relaxation

null

      answers WITH relaxation

AS COOMA 6955 150 3106
AS INVERELL 6939 98 2667
ID HUSEIN SASTRANEGARA 7361 148 2431
```

Query 6: near-to query with use-TAH “Knowledge Table”

Find at least 3 ports located near to the port 'BANGKOK'.

```
file: navy-port-near-to.qry

"select     CTRY_CODE, NAME, SEC_CLASS_CODE, LAT, LON
from  PORTS
where NAME near-to 'BANGKOK'
at-least 3"

transformed SQL query:
SELECT  PORTS.CTRY_CODE, PORTS.NAME, PORTS.SEC_CLASS_CODE, PORTS.LAT,
PORTS.LON
FROM    PORTS
WHERE   ( PORTS.LAT BETWEEN [11.75, 15.75]
        AND PORTS.LON BETWEEN [98.5, 102.5] )  USE-TAH "KNOWLEDGE_TABLE"

      answers WITHOUT relaxation

TH BANGKOK U 13.7500 100.5000

      answers WITH relaxation

TH BANGKOK U 13.7500 100.5000 0
TH KO SI CHANG HARBOR U 13.1667 100.8167 0.289487
TH AO UDOM U 13.1167 100.8833 0.322866
TH THUNG PRONG U 12.7000 100.8333 0.480477
TH CHUK SAMET U 12.6167 100.9167 0.526644
BM MERGUI U 12.4667 98.6000 1
```

Query 7: prefer & reject

Find at least 2 airfields whose runway width > 150 (if not enough answers, relax the width, excluding the range [100, 130]) and are also located at the elevation level with the preference order of 3000, 2000, 1000.

```
file: navy-air-prefer.qry

"select CTRY_CODE, AIRFIELD_NAME, RWWTH, ELEV
from  AIR_RWY
where       RWWTH > 150 reject ([100,130])
and   ELEV > prefer (3000, 2000, 1000)
at-least 2"

transformed SQL query:
SELECT  AIR_RWY.CTRY_CODE, AIR_RWY.AIRFIELD_NAME, AIR_RWY.RWWTH, AIR_RWY.ELEV
FROM    AIR_RWY
WHERE   AIR_RWY.RWWTH > 150
        AND ! (  NOT AIR_RWY.RWWTH BETWEEN [100, 130] )
        AND AIR_RWY.ELEV > 1000

      answers WITHOUT relaxation

null

      answers WITH relaxation

ID BADA 175 2625
ID BAUCAU 182 1726
```

Query 8: relax-order

Find airfields with elevation equal to 9000, runway length >6250, and Runway width > 100 relax elevation first and then runway length and width.

```
file: navy-air-relax-order.qry

"select     CTRY_CODE, AIRFIELD_NAME, ELEV, RWLTH, RWWTH
from  AIR_RWY
where (ELEV = 9000) label L1
and          (RWLTH > 6250 and RWWTH > 100) label L2
relax-order [L1 L2]"
```

Note: relax-order [L1 L2] means that "relax the condition L1 all the way up to the root. If still not enough, answers then relax the condition L2".

```
transformed SQL query:
SELECT  AIR_RWY.CTRY_CODE, AIR_RWY.AIRFIELD_NAME, AIR_RWY.ELEV,
AIR_RWY.RWLTH, AIR_RWY.RWWTH
FROM    AIR_RWY
WHERE   AIR_RWY.ELEV BETWEEN [2375, 9678]
        AND ( AIR_RWY.RWLTH > 6250
        AND AIR_RWY.RWWTH > 100 )
```

```
      answers WITHOUT relaxation
null
      answers WITH relaxation
```

Note: only ELEV is relaxed. The second query condition on runway length and width has not relaxed.

```
AS COOMA 3106 6955 150
ID HUSEIN SASTRANEGARA 2431 7361 148
```

When we don't use relax-order, the answers will be different. For instance, the following query is identical to the above one, with the exception of the "relax-order" and "label" construct, but the results are different as shown. This is because when there are multiple conditions that the system can relax, by default, the system at each step picks the best attribute to relax in terms of minimizing the relaxation error. This process continues until it has enough answers.

```
file: navy-air-relax-order-2.qry

"select     CTRY_CODE, AIRFIELD_NAME, ELEV, RWLTH, RWWTH
from  AIR_RWY
where (ELEV = 9000)
and         (RWLTH > 6250 and RWWTH > 100)"

transformed SQL query:
SELECT  AIR_RWY.CTRY_CODE, AIR_RWY.AIRFIELD_NAME, AIR_RWY.ELEV,
AIR_RWY.RWLTH, AIR_RWY.RWWTH
FROM    AIR_RWY
WHERE   AIR_RWY.ELEV BETWEEN [2375, 9678]
        AND AIR_RWY.RWLTH > 5868
        AND AIR_RWY.RWWTH > 83

      answers WITHOUT relaxation

null

      answers WITH relaxation

AS COOMA 3106 6955 150
AS INVERELL 2667 6939 98
ID HUSEIN SASTRANEGARA 2431 7361 148
```

Note: The system relaxes all the 3 conditions (elevation, runway length, runway width) based on a heuristic algorithm that chooses the best attribute to relax in each relaxation step.

E. Databases Used in the Demo

Table name: AIRFIELDS_INTL

<u>List of Attributes</u>

NAME
LAT
LON
SYMBOL
COLOR
CTRY_CODE
DESIG_ICAO
SEC_CLASS_CODE
CONF_LVL_CODE
AGENCY_COGNIZANT_CODE
AGENCY_SOURCE_CODE
DATE_ROW_SOURCE
DATE_ROW_LOAD
DATE_ROW_CHNG

Number of rows: 2465

Table name: AIR_RWY

<u>List of Attributes</u>

CTRY_CODE
AR_INDEX
AIRFIELD_NAME
LAT
LON
ELEV
SYM
RWLTH
RWWTH
RWY
COND
CODE

Number of rows: 86

Table name: CITY

<u>List of Attributes</u>

CTRY_CODE
CITY_NAME
DESIG
LAT
LON

```
AREA
UTM
JOG_NO
```

Number of rows: 1116

Table name: COUNTRY

List of Attributes

```
CY_CD
COUNTRY_NAME
```

Number of rows: 313

Table name: PORTS

List of Attributes

```
NAME
LAT
LON
SYMBOL
COLOR
OVERHEAD_LIMITS
DEPTH_CHANNEL
VESSEL_MAX_LENGTH
FIRST_PORT_ENTRY
US_REPRESENTATIVE
ETA_MESSAGE
COMPULSORY_PILOTAGE
AVAILABLE_PILOTAGE
LOCAL_ASSIST_PILOTAGE
ADVISABLE_PILOTAGE
TUGS_ASSIST
AIR_COMM
DEGAUSS
LONGSHORE_SERVICES
ELECT_SERVICES
STEAM_SERVICES
NAVIG_EQUIP_SERVICES
ELECT_REPAIR_SERVICES
PROVISIONS_SUPPLIES
WATER_SUPPLIES
FUEL_OIL_SUPPLIES
DIESEL_OIL_SUPPLIES
DECK_SUPPLIES
ENGINE_SUPPLIES
REPAIR
SIZE_DRYDOCK
CTRY_CODE
FORGN_CLNC_ADV_RQMT_DAYS
NUCLEAR_PIER
```

NUCLEAR_ANCHORAGE
APPROVED_REPAIR_PORT
MAJ_PORT
CONTAINING_AREA
FORWARD_DEPLOYED_HP
DEPTH_CARGO_PIER
SEC_CLASS_CODE
CONF_LVL_CODE
AGENCY_COGNIZANT_CODE
AGENCY_SOURCE_CODE
DATE_ROW_SOURCE
DATE_ROW_LOAD
DATE_ROW_CHNG

Number of rows: 4800

Table name: RUNWAY

List of Attributes

OCCURRENCE
WRLD_AREA_CD
INS_NUM_ID
RNWY_SUR_CHRTNG_CD
RNWY_SQNC_CD
RNWY_LDM
RNWY_WDM
RNWY_SUR_COMP_CD
RNWY_CND_CD
RNWY_ACFT_CAP_QN
RNWY_LCN_CD
RNWY_LCN_VLD_CD
RNWY_HDNG_LOW_END1
RNWY_HDNG_LOW_END2
RNWY_ARREST_CBL_CD
RNWY_ELVTN_LOW_CD
RNWY_OVRN_SUR_LOW_CD
RNWY_OVRN_LNG_LOW_CD
RNWY_OVRN_LCN_LOW_CD
RNWY_HDNG_HIGH_CD1
RNWY_HDNG_HIGH_CD2
RNWY_ARREST_BARR_CD
RNWY_ELVTN_HIGH_CD
RNWY_OVRN_SUR_HIGH_CD
RNWY_OVRN_LNG_HIGH_CD
RNWY_OVRN_LCN_HIGH_CD
RNWY_REMARKS_PTR
RUNWAY_LENGTH
RUNWAY_WIDTH

Number of rows: 18700

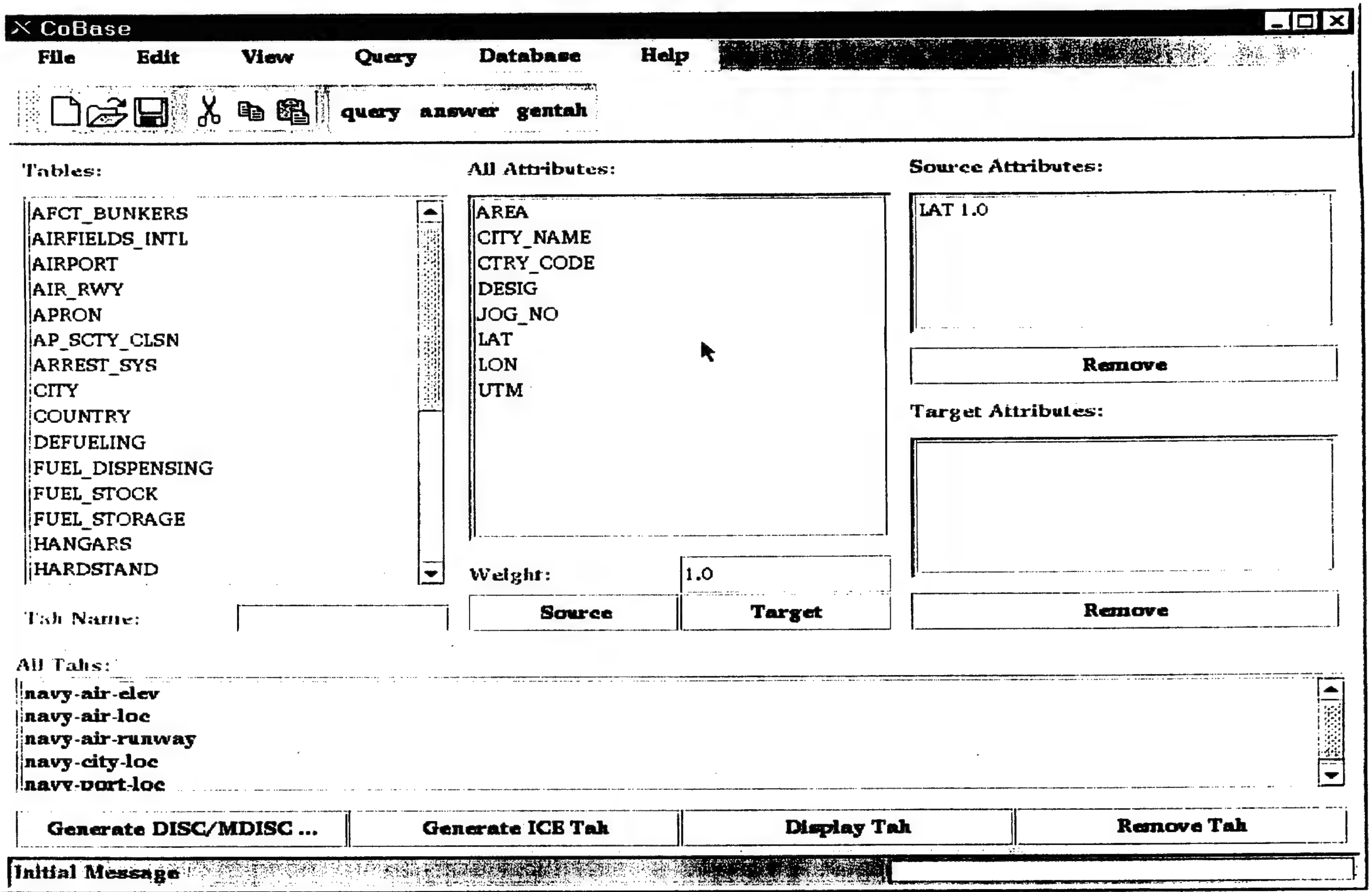
F. Demo Briefing

navy-city-similar-to.qry

User's query: **"Find at least 5 cities with country code, designation, latitude, longitude similar to Dili based on *latitude* and *longtitude***

CoSQL:

```
SELECT CTRY_CODE, CITY_NAME, DESIG,
       LAT, LON
FROM   CITY
WHERE  CITY_NAME similar-to 'Dili'
based-on ( (LAT,1.0),(LON,1.0) )
at-least 5
```



Tables:
AFCT_BUNKERS
AIRFIELDS_INTL
AIRPORT
AIR_RWY
APRON
AP_SCTY_CLSN
ARREST_SYS
CITY
COUNTRY
DEFUELING
FUEL_DISPENSING
FUEL_STOCK
FUEL_STORAGE
HANGARS
HARDSTAND

All Attributes:
AREA
CITY_NAME
CTRY_CODE
DESIG
JOG_NO
LAT
LON
UTM

Source Attributes:
LAT 1.0

Target Attributes:

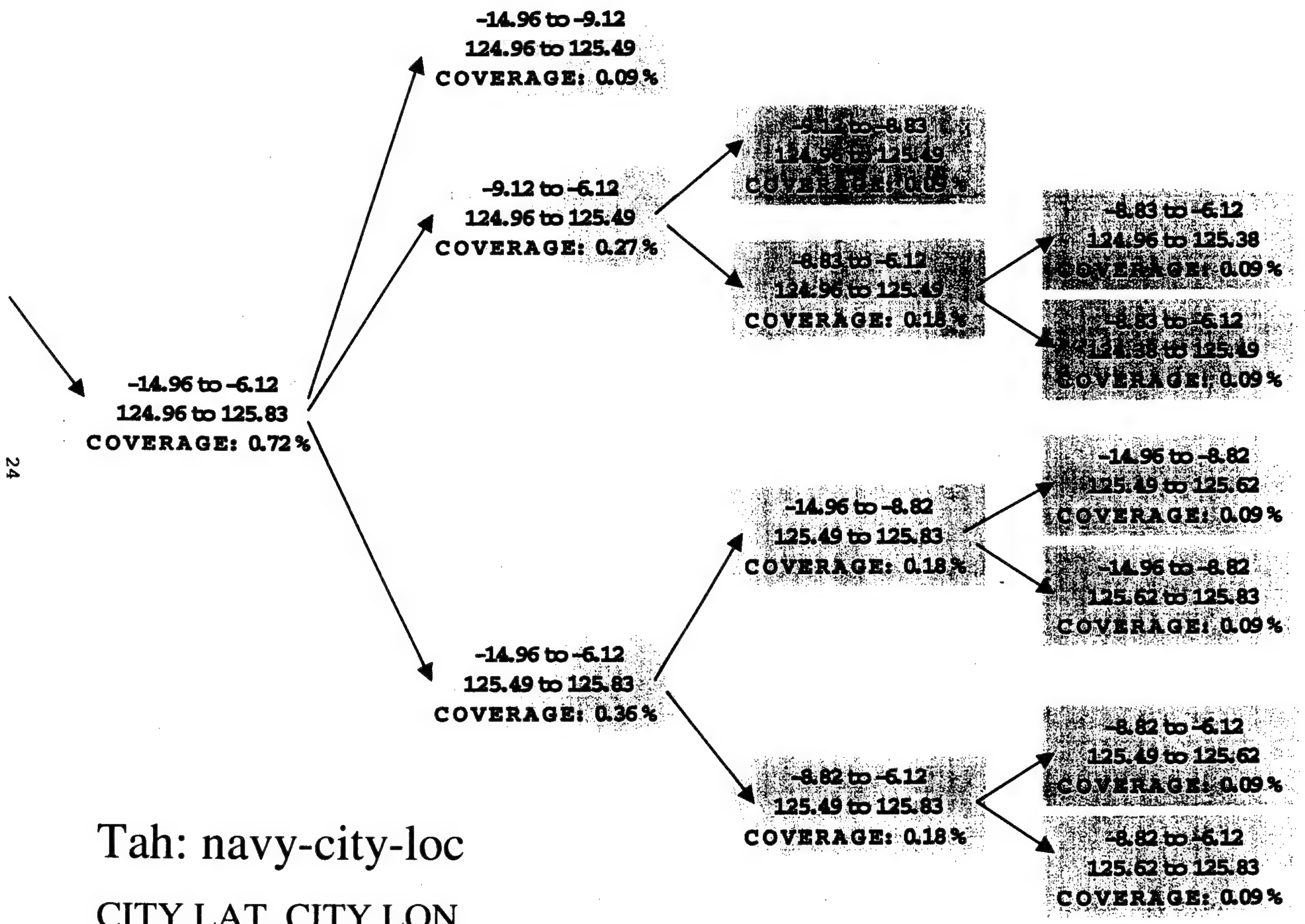
Weight: 1.0

Tah Name:

All Tahs:
navy-air-elev
navy-air-loc
navy-air-runway
navy-city-loc
navy-port-loc

Initial Message

Tah Generation GUI: User selects table and attribute(s) and gives tah name. The generated tah will be placed into the TahDirectory and the TahDirectoryFile that lists all available tahs will be updated.



Tah: navy-city-loc

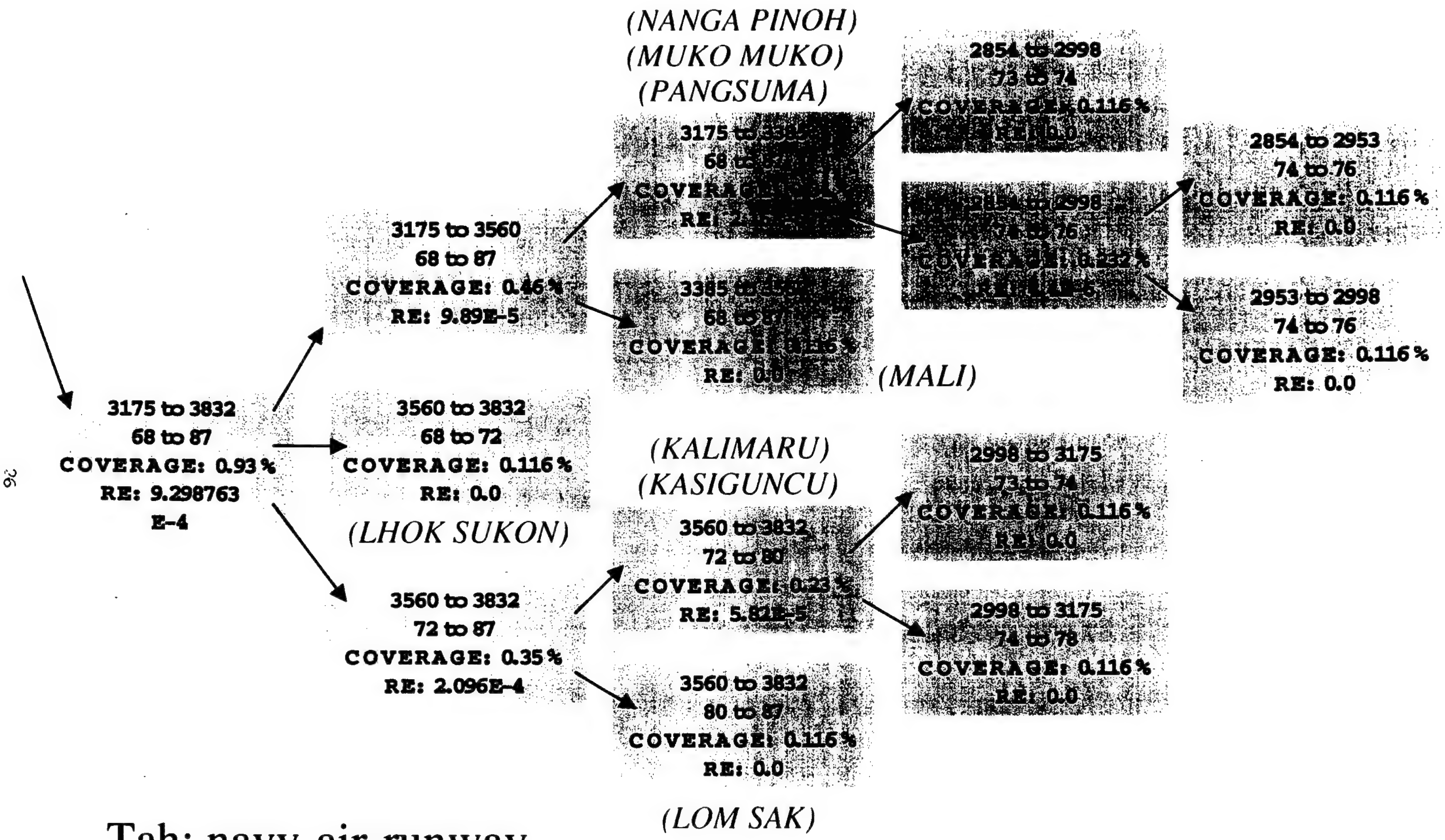
CITY.LAT, CITY.LON

navy-air-similar-to.qry

User's query: **"Find at least 7 airfields with country code, latitude, longitude similar to the Mali's airfield based on *runway length and runway width.*"**

CoSQL:

```
SELECT CTRY_CODE, AIRFIELD_NAME, LAT,
         LON
FROM   AIR_RWY
WHERE   AIRFIELD_NAME similar-to 'MALI'
based-on ( (RWLTH,1.0),(RWWTH,1.0) )
at-least 7
```



Tah: navy-air-runway

AIR_RWY.RWLTH, AIR_RWY.RWWTH

navy-air-relax-order.qry

User's query: **"Find airfields whose elevation are about 9000 and runway length and width is > 6250 and 100 approximately. When answers are not found, relax elevation first and then runway information second"**

CoSQL:

```
SELECT CTRY_CODE, AIRFIELD_NAME, ELEV,
          RWLTH, RWWTH
FROM AIR_RWY
WHERE (ELEV = 9000) label L1
AND (RWLTH > 6250 and RWWTH > 100) label L2
relax-order [L1 L2]
```

navy-air-prefer.qry

User's query: **"Find at 2 airfields whose runway width > 150 (if no enough answers, relax the width, excluding the range [100, 130]) and are also located at preferably 3000, 2000, 1000 elevation level"**

CoSQL:

```
SELECT CTRY_CODE, AIRFIELD_NAME,
            RWWTH, ELEV
FROM AIR_RWY
WHERE RWWTH > 150 reject ([100,130])
AND ELEV > prefer (3000, 2000, 1000)
at-least 2
```

navy-port-near-to.qry

User's query: **"Find at least 3 ports located near to the port Bangkok"**

CoSQL:

```
SELECT CTRY_CODE, NAME,
          SEC_CLASS_CODE, LAT, LON
FROM PORTS
WHERE NAME near-to 'BANGKOK'
at-least 3
```

navy-port-similar-to.qry

User's query: **"Find at least 3 ports with country code, section class code, latitude, longitude similar to Dover's port based on *latitude* and *longtitude***

CoSQL:

```
SELECT CTRY_CODE, NAME,
           SEC_CLASS_CODE, LAT, LON
FROM  PORTS
WHERE NAME similar-to 'DOVER'
based-on ( (LAT,1.0),(LON,1.0) )
at-least 3
```

navy-air-at-least.qry

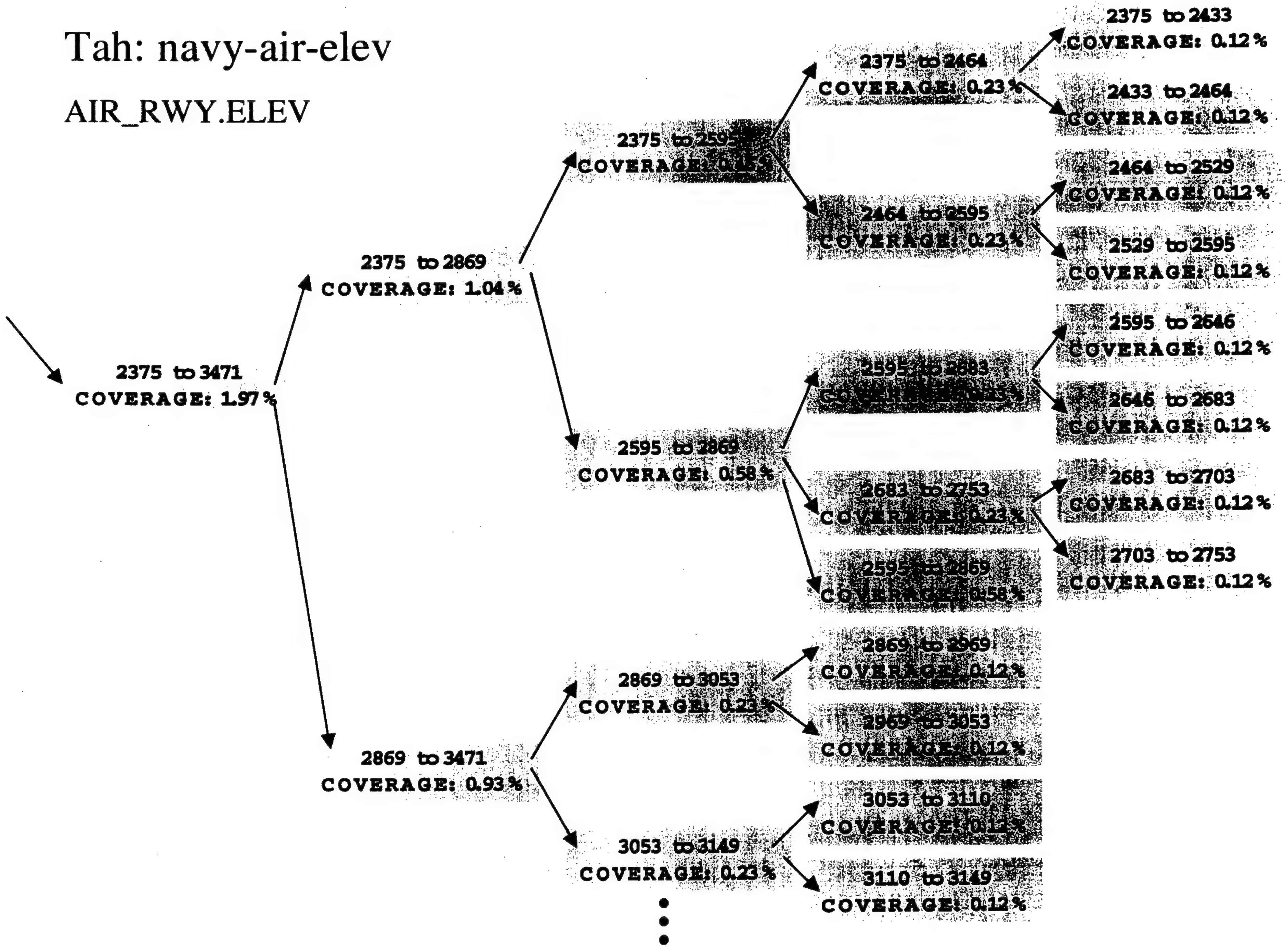
User's query: **"Find at least 3 airfields whose runway length > 6250 and runway width > 100 for landing a Boeing 707 type aircraft and are located at the 3000 elevation level approximately"**

CoSQL:

```
SELECT CTRY_CODE, AIRFIELD_NAME,
          RWLTH, RWWTH, ELEV
FROM  AIR_RWY
WHERE RWLTH > 6250 and RWWTH > 100
AND ELEV =~ 3000
at-least 3
```

Tah: navy-air-elev

AIR_RWY.ELEV



navy-air-runway.qry

User's query: **"Find at least 5 airfields with runway length, runway width where *runway length* > 3000 and *runway width* > 300"**

CoSQL:

```
SELECT AIRFIELD_NAME, RWLTH, RWWTH
FROM   AIR_RWY
WHERE  RWLTH > 3000 and RWWTH > 300
at-least 5
```

CoSQL: Approximation

User's query: Find airfields whose elevation is approximately 9000.

CoSQL:

```
select   CTRY_CODE, AIRFIELD_NAME, ELEV
from     AIR_RWY
where    ELEV =~ 9000
```

Note: =~ - relaxation primitive

G. Problems Observed During Demo and Proposed Solutions

The demo was successfully completed last September. During the demo, we also observed the following list of problems and proposed techniques to remedy them.

1) Resource balancing -- Since CoBase used CORBA as a communication between client and server there is some associated communication overhead. As a solution, we are in the process of replacing the COBRA connection between the GUI and CoBase with a Socket connection. Since a socket connection is of much lighter weight than a CORBA connection, this will solve the heavy memory or CPU resource problems.

2) Server start-up problem -- A shell script was written to automate the CoBase start-up task. We noted that when the CoBase server accidentally terminated, it had to restart manually. To resolve this problem, we are developing a light-weight CoBase daemon that: 1) monitors the CoBase server if it is alive and if not, start it properly, and 2) pass any input requests from the client to the CoBase server. This daemon will consist of a simple GUI for the system administration to start/stop the CoBase server. The communication between CoBase daemon and server will be done via socket connection.

3) For easier installation -- We are adding an InstallShield-based CoBase installation module. It will be supported on both NT and Solaris. Since the installation is based on a GUI, the installation task will be easier than before.

We have spent most of the remaining contract time repairing problems so that the future CoBase (CoBase 2000) will be both easier to install and distribute.

H. Appreciation Letters from Clients

-----Original Message-----
From: Gladys Kong [mailto:gkong@cs.ucla.edu]
Sent: Friday, August 13, 1999 9:39 AM
To: Wesley W Chu
Subject: FW: demo - comments

Dr. Chu,

This is Leah's comment after her demo on Wednesday (August 11).

Gladys

-----Original Message-----
From: Leah Wong [mailto:Wong@spawar.navy.mil]
Sent: Tuesday, August 10, 1999 7:23 AM
To: Gladys Kong
Cc: Wong@spawar.navy.mil
Subject: demo - comments

Gladys,

Thanks to your great work!

The demo went fine. The people could understand the essence of CoBase and were happy that CoBase could run on our machine and database. We are planning to show it to the other management groups soon ... I appreciate your feedback.

Leah

-----Original Message-----
From: Leah Wong [mailto:Wong@spawar.navy.mil]
Sent: Wednesday, September 29, 1999 3:12 AM
To: Wesley Chu; fmeng@cs.ucla.edu; hychiu@ucla.edu; dongwon@cs.ucla.edu
Subject: Thanks!

All,

I'd like to thank you all for your hard work and full support in recent days in preparing the CoBase demo via Ron's FIPA_SMART. All demos went fine yesterday. Although Ron didn't have time to format the returned answers from CoBase, I trust we have conveyed the idea of query relaxation to the sponsors.

I especially want to thank Professor Chu for allowing the team to work extra hours (Dongwon worked between vacations; Henry was always available to answer questions; Frank's help during his dissertation preparation) during the demo preparation. As you know, we waited until almost the last minute to see Ron's application working. I was immensely grateful while seeing Ron's demo worked for the first time yesterday during the review.

Best regards,
Leah

I. Install CoBase Demo at AFRL/Rome

We have also installed the CoBase Demo at the AFRL/Rome. We are given an account at AFRL/Rome so all the CoBase components were installed on the machine. Because of the COBRA version problem, we did run into some problems. These problems were resolved by replacing COBRA with socket server for the CoBase engine to communicate with its JAVA-based GUI.

A set of queries are provided to showcase CoBase relaxation technology, which includes relaxation operator and relaxation control. The relaxation operators includes "approximate", "similar to based on a set of attributes", and "ranking of relaxation answers" based on a set of pre-specified nearness measure. Relaxation control operations include "relaxation orders", "not relaxable", and "preference list."

J. References

[CC92] Chu, W. W. and Q. Chen, "Neighborhood and Associative Query Answering," Journal of Intelligent Information Systems, 1(3/4), 1992, pp. 355-382.

[CC94] Chu, W. W. and K. Chiang, "Abstraction of High-Level Concepts from Numerical Values in Databases," Proceedings of the AAAI Workshop on Knowledge Discovery in Databases, July 1994.

[CCH+96] Chu, W. W., K. Chiang, C. C. Hsu, and H. Yau, "An Error-Based Conceptual Clustering Method for Providing Approximate Query Answers," in Communications of the ACM, Vol. 39, No. 13, December 1996, http://www.acm.org/pubs/cacm/extension.

[CMB93] Chu, W. W., M. Merzbacher, and L. Berkovich, "The Design and Implementation of CoBase," SIGMOD '93, May 1993, pp. 517-522.

[CYC+96] Chu, W. W., H. Yang, K. Chiang, M. Minock, G. Chow, and C. Larson, "CoBase: A Scalable and Extensible Cooperative Information System," in Journal of Intelligent Information System, Vol. 6, 1996, pp. 223-260.

[MC93] Merzbacher, M. and W. W. Chu, "Pattern-Based Clustering for Database Attribute Values," AIII Workshop on Knowledge Discovery in Databases, Washington, D. C., July 1993.

☆U.S. GOVERNMENT PRINTING OFFICE: 2000-510-079-81269